Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 65 (2015) 442 – 449

Procedia Computer Science

International Conference on Communication, Management and Information Technology (ICCMIT 2015)

Addressing Challenges of Ultra Large Scale System on Requirements Engineering

Ahmed Safwat* and M.B.Senousy

Department of Information Technology
Sadat Academy for Management Sciences
Cairo, Egypt

Abstract

According to the growing evolution in complex systems and their integrations, Internet of things, communication, massive information flows and big data, a new type of systems has been raised to software engineers known as Ultra Large Scale (ULS) Systems. Hence, it requires dramatic change in all aspects of "Software Engineering" practices and their artifacts due to its unique characteristics.

Attendance of all software development members is impossible to meet in regular way and face-to-face, especially stakeholders from different national and organizational cultures. In addition, huge amount of data stored, number of integrations among software components and number of hardware elements. Those obstacles constrict design, development, testing, evolution, assessment and implementation phases of current software development methods

In this respect, ULS system that's considered as a system of systems, has gained considerable reflections on system development activities, as the scale is incomparable to the traditional systems since there are thousands of different stakeholders are involved in developing software, were each of them has different interests, complex and changing needs beside there are already new services are being integrated simultaneously to the current running ULS systems.

The scale of ULS systems makes a lot of challenges for Requirements Engineers (RE). As a result, the requirements engineering experts are working on some automatic tools to support requirement engineering activities to overcome many challenges.

This paper points to the limitations of the current RE practices for the challenges forced by ULS nature, and focus on the contributions of several approaches to overcome these difficulties in order to tackle unsolved areas of these solutions.

* Corresponding author.
Tel.: +201223484586.
E-mail address: ahmedsafwat@outlook.com

Peer-review under responsibility of Universal Society for Applied Research
doi:10.1016/j.procs.2015.09.116

As a result, the current approaches for ULS miss some RE essential practices related to find vital dependent requirements, and are not capable to measure the changes impact on ULS systems or other integrated legacy systems, in addition the requirements validation are somehow depended on the user ratings without solid approval from the stakeholders.

Peer-review under responsibility of Universal Society for Applied Research
Keywords: ULS; ULS Challenges; Requirements Engineering

1. Introduction

Computer-based systems are built for people and by people. Requirements engineering (RE) is essentially a social collaboration activity, in which involved stakeholders (e.g., customers and developers) have to closely work together to communicate, elicit, negotiate, define, confirm, and finally come up with the requirements (including functional and non-functional requirements) for the system to be implemented or upgraded [1].

As globalization is driving organizations to become more and more distributed, multi-site development is becoming a norm. With the increasing globalization in this industry, it is necessary to better prepare software development projects to manage work in distributed environments[2].

Especially in large-scale and distributed software projects, it is infeasible to organize personal meetings on a regular basis. In such scenarios, requirements are often defined in wiki-based forums which are receptive to the problems of information overload, redundancy, incompleteness of information, and diverging opinions of different stakeholders[3].

2. Requirement Engineering Practices

Requirements engineering covers several activities, including determining stakeholders, requirements elicitation, analysis, specification, verification and management as follows:

2.1. Stakeholder Analysis

Stakeholder is anyone is influencing or influenced by the system development and use the system either directly or indirectly[18]. Stakeholder's determination involves identifying the relevant stakeholders and prioritizing them based on their influence and interest in the project [4],

2.2. Requirement Elicitation

Requirement elicitation are the practices of discovering, reviewing, documenting and understanding the user's needs and constraints for a system[19]. And typical resulting artifacts are, for example, textual requirements descriptions, use cases, process diagrams and prototypical user interfaces

2.3. Requirement Analysis and Specifications

It's the process of refining the user's needs and constraints and documenting the user's needs and constraints clearly and precisely. In addition it includes the activities related to find the conflicts of interest and solve the problem in the requirements that contradict the organization and business rules.

2.4. Requirement Prioritization

Discovering the important requirements by interacting with the stakeholders and organize them in to most priority order [20]

2.5. Requirement Verification

Ensuring that the system requirements are complete, correct, consistent and clear is done as part of the requirements verification

3. Requirements Engineering in Small and Medium Software Development Project

The requirements for any system are the descriptions of the system services provided and the constraints on its operation. These requirements represent the customer needs for a system to serve a certain purpose, the process of finding out, analyzing, documenting, and checking these services and constraints is called requirements engineering (RE)[5]

Traditional development methodologies such as waterfall, spiral model and unified process are based on sequential series of steps such as requirements definition, coding, testing and deployment, always traditional methodologies require defining and documenting a stable set of requirements at the beginning of a project[6]

It is noticed that traditional developments give requirements documents very high weight and treat then as key deliverable for the requirement elicitation phase and believe that's possible to gather all of a customer's requirements, upfront, prior to writing any code and to sign off to proceed in the next software building activities, this process give very tough restriction on requirement changes

Some practitioners found the traditional methods posed difficulties in handling the requirements change even when change rates are relatively low, therefore, several experts have developed methodologies and practices in order to respond to mandatory changes they were facing, these methodologies are based on iterative enhancements, these techniques was introduced in 1975 and become known as agile methodologies[6]

4. The Limitations of Current Software Development Methodologies

Today's software development environments are heavily oriented toward traditional software development methodologies as they centralize the activities in a single organization and points of control.

Since the teams are first analyse requirement and write the specifications, and then proceed through detailed design, coding, testing and etc, whereas in ULS this cycle is unrealistic; Analysis and design methods must accommodate universal incompleteness, imperfection, uncertainty, and non-determinacy in the requirements and processes that arise throughout the system development and evolution[7]

Therefore we need a new paradigm of development supports the following activities:

- ULS System includes thousands, or even hundreds of thousands of stakeholders and it's obvious that attendance of all stakeholders is impossible
- The diversity of stakeholders comes from different cultures, concerns, policies, business processes those need ways to respond to their different, conflict, and changing requirements
- On the one hand, the relevant requirements for all the subsystems and the integration solution must be understood, updated, and transitioned into the architecture. Beside the more flexible the architecture is, the more adaptability of the ULS is for the changes in the operational environment[8]
- Requirements must be monitored and managed to ensure that no individual or organization can appreciably change the system without understanding, and perhaps getting approval from, the other participants[7].
- Distributed development activities over many organizations those require new methods for requirements compatibility, verifications, and problems detections.
- Continues evolution in an operational environment where the number of changes is very large and the period between design time and run time is blurring

- Abstracting the systems architectures, their interfaces and the context for evolving and adapting ULS systems
- Dynamic and rapid requirements response to maintain in situ ULS systems operational capabilities
- Automatic validation to support continues testing in real time and non-deterministic behavior[9]

5. Requirements Characteristics in ULS System

5.1. Incompleteness:

The absence of essential requirements after conducting requirement elicitations, and analysis practices. These requirements gathered customer's interactions, observations, or interviews. Incomplete requirements is one of important and crucial issues in RE as incorrect requirements major contributors to project budget overruns and schedule postponement[10]

5.2. Unknowable:

According to the scale and complexity of problems, in order to solve them by ULS systems mean that, in many cases, the requirements to be satisfied by the systems will not be adequately known until the system are in use, which means each solution will be provided will give a deeper understanding of what the problem is and lead to attempt for a solution[11]

5.3. Ever Changing

Requirements evolve over time, so the changes requests often referred to a change in requirements, changes might be issued from customers after requirements analysis or any other resource [10] such as developers, acquirers, suppliers, testers, or whoever is represented as a stakeholder, the ability to response to ever changing requirements in an decentralized way cannot possibly take all different purposes into account and manage them efficiently, or allow for rapid changes in response to immediate needs [7].

5.4. Conflicting

A system which has thousands of stakeholder is possible to have diversified community of stakeholders, each community has a lot of different groups, members and roles, these stakeholders groups have differing aspects for their needs and interests [12] and when multiple stakeholder participate in a discussions, requirements are often conflict [13]

5.5. Randomness

As ULS systems will comprehend so much functionality and therefore requirements gathering takes on a new complexion, there will be, in effect, randomness or uncertainty in requirements, in the specifications, and in the systems itself and it may not be possible to determine reliably where the problem comes [7]

5.6. Diversity

ULS systems' projects are anticipated to be highly complex and to involve thousands, or even hundreds of thousands of stakeholders [14] who develop and use these systems, this mixture comes with different cultures, languages, geographical zones, ages, communications infrastructures, individual educational capabilities and levels, and technical knowledge. Consequently, it also means a much longer and more complicated set of communication channels for requirements knowledge to travel between these stakeholder groups.[15]

6. Proposed Solutions and Approaches for ULS Requirements Engineering Challenges

Several researches and tools have been developed for supporting various requirements engineering challenges for ULS, and suggested some tools and techniques this paper made a wide outlook for the recent work done for each challenge along with RE activities and reached to the following analysis:

Table 1. Overview of RE challenges and recommendation approaches

Requirement Activity	Model and Tools
Stakeholders Analysis	StakeSource Using Social Networks and Crowdsourcing[4]
Requirements Elicitation	Data Mining and Recommender Systems and k-Nearest Neighbor algorithm: kNN is used to identify like-minded users with similar rating histories [17]
Requirements Classification	StakeRare, where the requirements classification are determined by the requirement engineer and may be modified during the elicitation process [21]
Requirements Prioritization	StakeSource2, stakeholder's ratings on the requirements and their influence in the project. [16]
Requirements Prediction	Collaborative filtering to predict other requirements[12]
Finding Requirements Conflict	StakeSource2.0 which highlights stakeholders with conflicting preferences for requirements[16]
Managing Requirements uncertainty	Using MAVO to express uncertainty reduction in RE models[22]

6.1. Stakeholders Analysis

Soo, Damian, and Finkelstein established StakeSource2.0, a web-based tool that uses social networks and collaborative filtering, a "crowdsourcing" approach, to identify and prioritise stakeholders and their requirements. It proposes a shift from traditional methods for requirements analysis where system analysts conduct requirements elicitation, to a crowdsourcing approach where all stakeholders have a say [16].

6.2. Requirements Elicitations

Jane, and Bamshad described a proposed approach that utilizes data mining and recommender systems to scale up the requirements elicitation[13], whereas Mulla and Girase utilized another approach uses social networks and collaborative filtering for requirement elicitation for large scale projects to identify requirements[17]

6.3. Requirements Prioritization and Prediction

Through using StakeSource2.0, it prioritize requirements by asking stakeholders to rate requirements, and prioritises the requirements using their ratings weighted by their priority in the social network, in addition, it can predicts other requirements using collaborative filtering techniques by collecting preference information from many users and recommending discussion forums of interest for stakeholders.[16]

6.4. Requirements Classification

The requirements were grouped under their respective project objectives. Specific requirements were classified into their respective requirements. [21]

6.5. Requirements Conflict

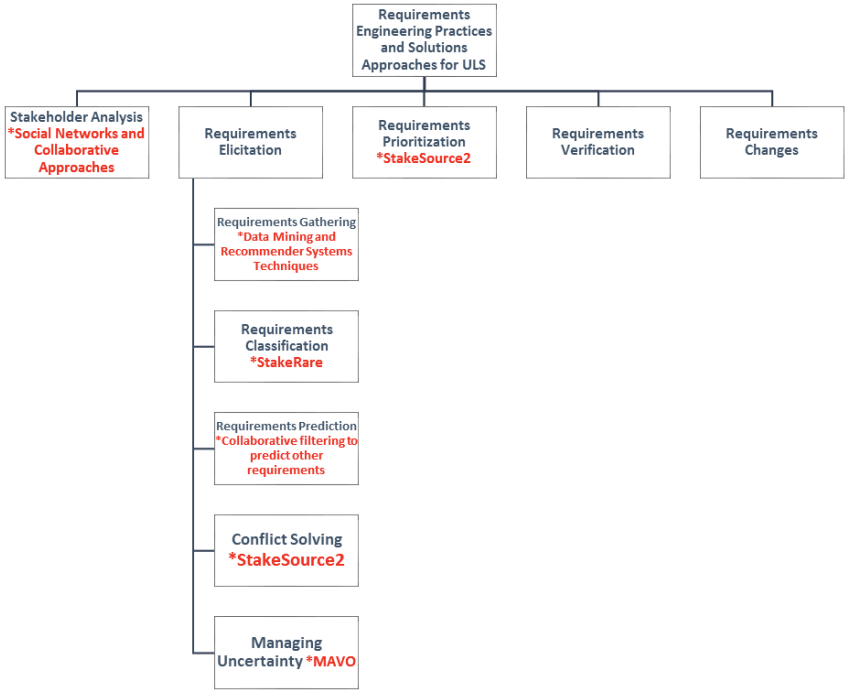
All developed tools such as StakeSource2.0 used noticing gap among the rating requirements those assigned by stakeholder, from this point it shows the conflicting preferences, and highlights stakeholders with conflicting preferences for requirements and reveals their position in the social network. [16]

6.6. Managing Requirements Uncertainty

MAVO is a formal approach developed by Rick with his team for expressing and reasoning with model uncertainty and they expanded their work to answer on some questions about uncertainty in RE [22].

The following hierarchy presenting a wide view for the previous solutions as highlighted in blue for each RE practices and, other practices should be tackled to overcome the challenges of ULS for RE

Figure 1. Wide View for RE Practices and their proposed solution for ULS



7. Conclusion

In large and ultra large scale system, it's noticed that using the traditional requirements engineering methods hat depend on interactive communications channels between the analysts and systems' stakeholders are inadequate, and

the produced solutions approaches are oriented to automatic tools using social networks, recommender system and data mining techniques.

Still the new techniques neglect some essential practices attached to requirements engineering as they handle the requirements flows and miss catching other dependents requirements, those had been used to cover by brainstorming, workshops and interviews methods. In addition, changes impact analysis is absent for analysing changes effect on other requirements and changes in development phase.

Another problem related to find essential requirements needed to complete required business system processes and rules as most of tools depend on users to fill needed requirements without direction. Requirements validation is a question mark as well from the perspective of how will the requirements and the changes be confirmed against the stakeholder needs and verified as consistent.

Those problems need more attention from the Software Engineering Scientists as we still in the middle of road toward dynamic requirements engineering for ULS systems.

8. Future Work

To address the drawbacks, future work should find ways to measure and analyze impact of changes on all the developed requirements those ways could utilize data mining techniques or cognitive computing.

In fact, finding more dependent requirements those are not transferred or being tacit stakeholders such as business rules, process, policies or laws requires much work in using knowledge management and text mining techniques to extract such requirements

Still, we seek to find better ways to get final stakeholder confirmation for the analyzed requirements, in order to travel the approved requirement to distributed development units

References

1. Peng Liang. Paris Avgeriou, Keqing He, Lai Xu, *From Collective Knowledge to Intelligence: Pre-Requirements Analysis of Large a Complex Systems,* Web2SE'10, May 4, 2010
2. Tanmaya Kumar, Dillip Kumar, Gopa Krishna, *Towards Large Scale Software Project Development and Management,* IOSR Journal of Computer Engineering (IOSR-JCE) Volume 8, Issue 6 (Jan – Feb 2013) p: 20-35
3. A. Felfering, G. Ninaus, H. Grabner, F. Reinfrank, L. Weninger, D. Pagano, W. Maleej, *An Overview of Recommender Systems in Requirements Engineering,* Chapter 14 of *Managing Requirements Engineering,* p: 319, 2013
4. Soo Ling Lim, Cornelius Ncube, *Social Networks and Crowdsourcing for Stakeholder Analysis in System of Systems Projects*, Proc. of the 2013 8th International Conference on System of Systems Engineering, Maui, Hawaii, USA - June 2-6, 2013
5. Ian Sommerville. *Software Engineering*, 9th Edition, 2011, Addison-Wesley, 83 – 87
6. M.A.Awad. *A Comparison between Agile and Traditional Software Development Methodologies*. The University of Western Australia, 2005. P.3
7. Feiler, P., Gabriel, R.P., Goodenough, J., Linger, R., Longstaff, T., Kazman, R., Klein, M., Northrop, L., Schmidt, D., Sullivan, K., Wallnau, K.: *Ultra-large-scale systems: The software challenge of the future*. Technical report, Software Engineering Institute. (2006)
8. Barry Demchak, Vina Ermagan, Claudiu Farcas, Emilia Farcas, Ingolf H. Krüger, Massimiliano Menarini, *Rich Services: Addressing Challenges of Ultra-Large-Scale Software-Intensive Systems*, University of California, San Diego
9. Ian Sommerville, Dave Cliff, Radu Calinescu, Justin Keen, Tim Kelly, Marta Kwiatkowska, John McDermid and Richard Paige, *Large-scale Complex IT Systems*, Magzine Communications of the ACM Volume 55, Issue 7, 2012, Pages 71-77
10. A.O. Elfaki, *Automated Verification of Variability Model Using First-Order Logic,* Chapter 12, Managing Requirements Engineering, P 263
11. Walid Maleej, Anil Kumar, Managing Requirements Knowledge, Springer Heidelberg 2013, p.19
12. Soo Ling Lim, and Anthony Finkelste, *StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation,*IEEE Transactions On Software Engineering, Manuscript ID
13. Shariful Islam Majumdar, Md. Saidur Rahman, Md. Mijanur Rahman, *Throny Issues of Stakeholder Identification and Prioterization in Requirements Engineering Process*, IOSR Journal of Computer Engineering Volume15, Issue 5 (Nov. - Dec. 2013), PP 73-78
14. Jane Cleland-Huang, Bamshad Mobasher, *Using Data Mining and Recommender Systems to Scale up the Requirements Process.*ULSSIS '08 Proceedings of the 2nd international workshop on Ultra-large-scale software-intensive systems p: 3-6
15. V.Dheepa, D.John Aravindhar, C.Vijayalakshmi, *A Novel Method for Large Scale Requirement Elicitation*, International Journal of Engineering and Innovative Technology (IJEIT) Volume 2, Issue 7, January 2013
16. Soo Ling Him, Daniela Damin, Anthony Finkelstein, *StakeSource2.0: Using Social Networks of Stakeholders to Identify and Prioritise Requirements*, ICSE'11, May 2011
17. Niolofar Mulla and Sheetel Girase, *A New Approach to Requirement Elicitaion Based on Stakeholder Recommendation and Collaborative Filtering*, International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.3, May 2012
18. Helen Sharp, Anthony Finkelstein and Galal Galal, *Stakeholder Identification in the Requirements Engineering Process*, *http://discovery.ucl.ac.uk/744/1/1.7_stake.pdf*
19. A Framework for Software Product Line Practice, Version 5.0, http://www.sei.cmu.edu/productlines/frame_report/req_eng.htm
20. Sai Ganesh. Gunda, *Requirements Engineering: Elicitation Techniques,* University West, 2008:PR003
21. Soo Ling Lim, and Anthony Finkelstein, *StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation,* IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, MANUSCRIPT ID
22. Rick Salay, Marsha Chechik, Jennifer Horkoff, Alessio Di Sandro: *Managing Requirements uncertainty with partial models,* Springer-Verlag London 2013